IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Applicant:	SHEN et al.	Patent Application	
Application No.:	10/687,997	Group Art Unit:	2157
Filed:	October 17, 2003	Examiner:	El Chanti, Hussein A.

For: SHARED RUNNING-BUFFER-BASED CACHING SYSTEM

APPEAL BRIEF

Table of Contents

	Page
Real Party in Interest	1
Related Appeals and Interferences	2
Status of Claims	3
Status of Amendments	4
Summary of Claimed Subject Matter	5
Grounds of Rejection to Be Reviewed on Appeal	8
Argument	9
Conclusion	16
Appendix - Clean Copy of Claims on Appeal	17
Appendix – Evidence Appendix	22
Appendix – Related Proceedings Appendix	23

I. Real Party in Interest

The assignee of the present invention is Hewlett-Packard Development Company, L.P.

II. Related Appeals and Interferences

There are no related appeals or interferences known to the Appellants.

III. Status of Claims

Claims 1-15 are pending. Claims 1-7 and 9-14 are rejected. Claims 8 and 15 are objected to. This Appeal involves Claims 1-7 and 9-14.

IV. Status of Amendments

All proposed amendments have been entered. An amendment subsequent to the Final Action has not been filed.

V. Summary of Claimed Subject Matter-

Independent Claims 1, 2 and 9 of the instant application pertain to embodiments of the present invention.

As recited in Claim 1, "[a] network proxy server" is described. This embodiment is depicted at least in Fig. 1. As shown in Fig. 1, proxy server 114 receives requests, e.g., requests 120, 122 and 124, from a client, e.g., clients 116-118 (page 5, lines 3-4). "Multiple, moving-window buffers are used to service content requests of the server by various independent clients" (page 3, lines 18-19). "A cache memory 108 includes many recirculating buffers, as represented by a first buffer 110 and a second buffer 112. A proxy server 114 hosts the cache memory 108 and off-loads work from server 101. The buffers 110 and 112 receive copies of the content passing through from the server to any of clients 116-118." (page 4, lines 29-34). "A first request for content is delivered by the server through the proxy to the requesting client. The content is simultaneously duplicated to a first circulating buffer. Once the buffer fills, the earlier parts are automatically deleted. The buffer therefore holds a most-recently delivered window of content. If a second request for the same content comes in, a check is made to see if the start of the content is still in the first buffer. If it is, the content is delivered from the first buffer. Otherwise, a second buffer is opened and both buffers are used to deliver what they can simultaneously. Such process can open up third and fourth buffers depending on the size of the content, the size of the buffers, and the respective timing of requests" (page 3, lines 19-32).

As recited in Claim 2, "[a] method of delivering objects from servers to clients" is described. This embodiment is depicted at least in Figure 6. "A method embodiment of the present invention comprises receiving a first request for an content object" (page 11, lines 28-29). "A request [601] is received from a client for an content object" (page 12, lines 15-16). "The

new buffer is then allocated [610] to store a sliding window of data from the requested content object. A process [612] retrieve the content object as a datastream and inserts the datastream into the newly allocated buffer while at the same time delivering the datastream to the client [614]." (page 12, lines 28-33). "When the initial buffer is filled, data is deleted from the start of the datastream while continuing to insert retrieved data into the buffer. The buffer contains a moving window of the retrieved data. A second request is received. If such is received while the start of the datastream is in the initial buffer, the content object is served directly from the initial buffer. If the second request is received after the start point has been deleted from the initial buffer, the portion of the content object that has been deleted from the initial buffer is fetched, commencing from the start point. Such is delivered simultaneously with other parts of the content object from the initial buffer" (page 11, line 34, through page 12, line 12).

As recited in Claim 9, "[a] computer-implemented method of shared running-buffer-based caching" is described. This embodiment is depicted at least in Figure 6. "A method embodiment of the present invention comprises receiving a first request for an content object" (page 11, lines 28-29). "A request [601] is received from a client for an content object" (page 12, lines 15-16). "The new buffer is then allocated [610] to store a sliding window of data from the requested content object. A process [612] retrieve the content object as a datastream and inserts the datastream into the newly allocated buffer while at the same time delivering the datastream to the client [614]." (page 12, lines 28-33). "When the initial buffer is filled, data is deleted from the start of the datastream while continuing to insert retrieved data into the buffer. The buffer contains a moving window of the retrieved data. A second request is received. If such is received while the start of the datastream is in the initial buffer, the content object is served directly from the initial buffer. If the second request is received after the start point has been deleted from the initial buffer, the portion of the content object that has been deleted from the initial buffer is

fetched, commencing from the start point. Such is delivered simultaneously with other parts of the content object from the initial buffer" (page 11, line 34, through page 12, line 12).

VI. Grounds of Rejection to Be Reviewed on Appeal

1. Claims 1-7 and 9-14 are rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,933,603 by Vahalia et al. (referred to hereinafter as "Vahalia").

VII. Argument

1. Whether Claims 1-7 and 9-14 are anticipated under 35 U.S.C. § 102(b) by Vahalia.

According to the Final Office Action mailed January 25, 2008, Claims 1-7 and 9-14 are rejected under 35 U.S.C. §102(b) as being anticipated by Vahalia. Appellants have reviewed Vahalia and respectfully submit that the embodiments as recited in Claims 1-7 and 9-14 are not anticipated by Vahalia for at least the following rationale.

MPEP §2131 provides:

> "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). ... "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

Appellants respectfully submit that the rejection of the Claims is improper as the rejection of Claims 1-7 and 9-14 does not satisfy the requirements of a *prima facie* case of anticipation as claim embodiments are not met by Vahalia. Appellants respectfully submit that Vahalia does not teach or suggest each element of the claimed embodiments in the manner set forth in independent Claims 1, 2 and 9.

A. Independent Claim 1

Independent Claim 1, recites (emphasis added):

> A network proxy server comprising:
> a network connection configured to receive content-object requests generated by a plurality of clients, said content-object requests requesting a content-object from a server;

a plurality of moving-window buffers coupled with said network connection, said plurality of moving-window buffers being configured to service said content-object requests; and

first and second content buffers coupled with said network connection, said first content buffer being configured to duplicate a first portion of a content passing from said server to said plurality of clients, cache said first portion, and provide said first portion to a subsequent client in response to a request for said first portion, and said second content buffer being configured to duplicate a second portion of said content and cache said second portion, and wherein said first and second content buffers are further configured to simultaneously provide said first and second portions of said content to said subsequent client in response to a request for said first and second portions.

Appellants respectfully submit that Vahalia fails to disclose each and every element of Claim 1, arranged as in the claim. Appellants understand Vahalia to teach "maintaining and dynamically allocating sliding windows of video data in the random access memories of the stream server computers" (Abstract). Appellants further understand Vahalia to teach "[w]hen a stream server computer becomes overloaded or nearly overloaded by client requests, the reserve memory in another stream server computer is allocated to store a duplicate of the original portion of the data set in the stream server that is overloaded" (emphasis added; col. 3, lines 28-32). Appellants do not understand Vahalia to anticipate the claimed embodiment as recited in Claim 1.

Appellants understand Vahalia to disclose a system for allocating server RAM to a movie, whereby each stream server of a plurality of stream servers acts as an individual sliding window. In particular, with reference to Fig. 16, Vahalia recites "[p]referably the block of data in the RAM of each of the four stream servers 91, 92, 93 and 94 is a sliding "window" into the movie (emphasis added; col. 23, lines 1-3). Moreover, "if a client would request a stop, fast-forward, or fast-reverse operation, it may be necessary to re-allocate a network client to a different stream server PC. In these cases, however, some delay would be

acceptable before the client could resume the viewing of the movie. If a stop, fast-forward or fast-reverse operation takes the client's viewing out of the window, then the client's continued viewing of the movie can be treated similar to a new request" (emphasis added; col. 23, lines 10-18)

In particular, Appellants respectfully submit that Vahalia does not disclose a proxy server comprising "a plurality of moving-window buffers" or "a first and second buffer" as claimed. In contrast, Appellants understand Vahalia to disclose a system for utilizing the RAM of a stream server as a single sliding window. Specifically, as presented above, Appellants submit that Vahalia disclose that "[w]hen a stream server computer becomes overloaded or nearly overloaded by client requests, the reserve memory in another stream server computer is allocated to store a duplicate of the original portion of the data set in the stream server that is overloaded" (emphasis added; col. 3, lines 28-32).

In summary, Appellants respectfully submit that the rejections of the Claims are improper as the rejection of Claim 1 does not satisfy the requirements of a *prima facie* case of anticipation as each and every element as set forth in the claim arranged as in the claim are not found in Vahalia. In view of Vahalia not satisfying the requirements of a *prima facie* case of anticipation, Appellants respectfully submit that independent Claim 1 overcomes the rejection under 35 U.S.C. § 102(b), and that this claims is thus in a condition for allowance.

B. Independent Claims 2 and 9

Appellants respectfully submit that independent Claim 2 recites that an embodiment of the present invention is directed to (emphasis added):

A method of delivering objects from servers to clients, said method comprising:

receiving a first request for a content object from a first client;

allocating a first running buffer;

retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the datastream to the first client;

when the first running buffer is filled, deleting data from the start point of the datastream while continuing to insert retrieved data into the first running buffer so that the first running buffer contains a moving window of the retrieved data;

receiving a second request for the content object from a second client;

if the second request is received while the start point of the datastream is still in the first running buffer, serving the content object directly from the first running buffer; and

if the second request is received after the start point has been deleted from the first running buffer, retrieving a portion of the content object that has been deleted from the first running buffer, commencing from the start point, and delivering the datastream while simultaneously delivering a different part of the content object from the first running buffer.

Independent Claim 9 includes similar embodiments. Claims 3-7 that depend from independent Claim 2 and Claims 10-14 that depend from independent Claim 9 also include these embodiments.

Appellants respectfully submit that Vahalia fails to disclose "retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the datastream to the first client" (emphasis added) as claimed.

Appellants understand Vahalia to teach "[w]hen a stream server computer becomes overloaded or nearly overloaded by client requests, the reserve memory in another stream server computer is allocated to store a duplicate of the original portion of the data set in the

stream server that is overloaded" (emphasis added; col. 3, lines 28-32). Appellants do not understand Vahalia to anticipate the claimed embodiment as recited in Claims 2 and 9.

With reference to Figs. 9 and 10 of Vahalia, a prefetch task for scheduling the transmission of video prefetch commands from a stream server is described. Vahalia recites "[i]n FIG. 9, video data are transmitted isochronously to a first network client from a buffer 91 in random access memory (RAM) in a first one of the stream servers (21 in FIG. 2). The buffer 91 is filled by data fetched from the cache 41 of the integrated cached disk array (23 in FIG. 2). The cache 41 is filled by data prefetched from the disk array 47" (emphasis added; col. 18, lines 17-23).

Furthermore, with reference to Fig. 17, Vahalia discloses "a procedure used in the admission control program for servicing client requests from the network and implementing an admission policy for client requests for a popular movie" (col. 23, lines 55-58). Vahalia recites "[i]f in step 174 the new request does not fall in a RAM window of the requested movie in the indexed stream server PC, or in step 175 the indexed stream server PC does not have sufficient resources to handle the request, then execution branches to step 177. In step 177, the admission control program checks whether all of the valid or operable stream server PCs have been inspected in the process of searching for a stream server PC than can satisfy the client request" (col. 24, lines 30-37). "If an unallocated RAM window is not found, then execution branches to step 180 to reject the client request. Otherwise, in step 181, a server window RAM is assigned to the movie, and a task is initiated to load this server window RAM with duplicate movie data fetched from the ICDA" (col. 24, lines 50-55). "Then in

step 176, the client request is accepted by assigning it to the indexed stream server PC, which has the server window RAM assigned in step 181" (col. 24, lines 60-63).

In particular, Vahalia recites "a set of RAM windows in the RAM 91, 92, 93, 94 of the stream server PCs (21 in FIG. 2) are allocated and loaded with the data for each popular movie before the client requests for the movie are received, so that when a client request for the movie is received, the client can be immediately supplied with a video stream starting at any desired time or position in the movie. In step 181, a new RAM window is allocated and loaded with data when a new client request cannot be serviced from an existing RAM window because the resources of the stream server PC having the existing RAM window are used up in the servicing of prior client requests" (emphasis added; col. 24, line 64, through col. 25, line 9).

In particular, Appellants respectfully submit that Vahalia does not disclose a proxy server comprising "retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the datastream to the first client" (emphasis added) as claimed. In contrast, as presented above, Appellants understand Vahalia to disclose a system in which a RAM window does not serve data to a client until it is filled. Specifically, as presented above, Appellants submit that Vahalia disclose that "a set of RAM windows in the RAM 91, 92, 93, 94 of the stream server PCs (21 in FIG. 2) are allocated and loaded with the data for each popular movie before the client requests for the movie are received, so that when a client request for the movie is received, the client can be immediately supplied with a video stream starting at any desired time or position in the movie" (emphasis added; col. 24, line 64, through col. 25, line 4).

In summary, Appellants respectfully submit that the rejections of the Claims are improper as the rejection of Claims 2-7 and 9-14 does not satisfy the requirements of a *prima facie* case of anticipation as each and every element as set forth in the claim arranged as in the claim are not found in Vahalia.

In view of Vahalia not satisfying the requirements of a *prima facie* case of anticipation, Appellants respectfully submit that independent Claims 2 and 9 overcome the rejection under 35 U.S.C. § 102(b), and that these claims are thus in a condition for allowance. Therefore, Appellants respectfully submit that Vahalia also does not teach or suggest the additional claimed features as recited in Claims 3-7 that depend from independent Claim 2 and Claims 10-14 that depend from independent Claim 9. Therefore, Appellants respectfully submit that Claims 3-7 and 9-14 also overcome the rejection under 35 U.S.C. § 102(b), and are in a condition for allowance as being dependent on an allowable base claim.

Conclusion

Appellants believe that pending Claims 1-7 and 9-14 are not anticipated by Vahalia. Therefore, Appellants respectfully submit that the rejections of the Claims are improper as the rejection of Claims 1-7 and 9-14 does not satisfy the requirements of a *prima facie* case of anticipation. Accordingly, Appellants respectfully submit that the rejections of Claims 1-7 and 9-14 under 35 U.S.C. §102(b) is improper and should be reversed.

The Appellants wish to encourage the Examiner or a member of the Board of Patent Appeals to telephone the Appellants' undersigned representative if it is felt that a telephone conference could expedite prosecution.

Respectfully submitted,
WAGNER BLECHER LLP

Dated: 05/23/2008

/John P. Wagner, Jr./
John P. Wagner, Jr.
Registration No. 35,398
123 Westridge Drive
Watsonville, CA 95076
(408) 377-0500

VIII. Appendix - Clean Copy of Claims on Appeal

1. A network proxy server comprising:

a network connection configured to receive content-object requests generated by a plurality of clients, said content-object requests requesting a content-object from a server;

a plurality of moving-window buffers coupled with said network connection, said plurality of moving-window buffers being configured to service said content-object requests; and

first and second content buffers coupled with said network connection, said first content buffer being configured to duplicate a first portion of a content passing from said server to said plurality of clients, cache said first portion, and provide said first portion to a subsequent client in response to a request for said first portion, and said second content buffer being configured to duplicate a second portion of said content and cache said second portion, and wherein said first and second content buffers are further configured to simultaneously provide said first and second portions of said content to said subsequent client in response to a request for said first and second portions.

2. A method of delivering objects from servers to clients, said method comprising:

receiving a first request for a content object from a first client;

allocating a first running buffer;

retrieving the content object as a datastream having a start point and inserting the datastream into the first running buffer while delivering the datastream to the first client;

when the first running buffer is filled, deleting data from the start point of the datastream while continuing to insert retrieved data into the first running buffer so that the first running buffer contains a moving window of the retrieved data;

receiving a second request for the content object from a second client;

if the second request is received while the start point of the datastream is still in the first running buffer, serving the content object directly from the first running buffer; and

if the second request is received after the start point has been deleted from the first running buffer, retrieving a portion of the content object that has been deleted from the first running buffer, commencing from the start point, and delivering the datastream while simultaneously delivering a different part of the content object from the first running buffer.

3. The method of claim 2, further comprising:

allocating a second running buffer when the second request is received after the start point of the datastream has been deleted from the first running buffer and inserting a portion of the content object not in the first running buffer into the second running buffer while delivering the datastream.

4. The method of claim 3, further comprising:

receiving a third request for the content object after the second running buffer has been allocated;

checking whether the start point is cached in an existing running buffer;

if the start point is cached in an existing running buffer, serving the content object as a datastream from each of the running buffers simultaneously;

if the start point is not cached in an existing running buffer, allocating a third running buffer; and

retrieving a portion of the content object not in an existing running buffer as a datastream and inserting the datastream into the third running buffer while delivering the datastream and simultaneously delivering a different part of the content object from other existing running buffers.

5. The method system of claim 2, further comprising:

determining a size of the first buffer as a proportion of an advertised length of the content object.

6. The method of claim 2, further comprising:

modifying a size of the first buffer or another buffer in response to an analysis of a frequency of requests for the content object in order to optimize an allocation of a memory.

7. The method of claim 2, further comprising:

prior to allocating the first buffer, applying a replacement algorithm to reclaim buffers from less frequently requested objects.

9. A computer-implemented method of shared running-buffer-based caching, said computer-implemented method comprising:

receiving a first request for a content object;

allocating a first running buffer;

retrieving the content object as a datastream having a start point and-inserting the datastream into the first running buffer while delivering the datastream;

when the first running buffer is filled, deleting data from the start point of the datastream while continuing to insert retrieved data into the first running buffer, so that the first running buffer contains a moving window of the retrieved data;

receiving a second request for the content object;

if the second request is received while the start point of the datastream is in the first running buffer, serving the content object directly from the first running buffer;

if the second request is received after the start point has been deleted from the first running buffer:

retrieving a portion of the content object that has been deleted from the first running buffer, commencing from the start point, and delivering the datastream while simultaneously delivering a different part of the content object as a datastream from the first running buffer.

10. The computer-implemented method of claim 9, further comprising:

if the second request is received after the start point of the datastream has been deleted from the first running buffer, allocating a second running buffer and inserting the datastream representing a portion of the content object not in the first running buffer into the second running buffer while delivering the datastream.

11. The computer-implemented method of claim 9, further comprising:

receiving a third request for the content object after the second running buffer has been allocated;

checking whether the start point is cached in an existing running buffer;

if the start point is cached in an existing running buffer, serving the content object as a datastream from each of the running buffers simultaneously;

if the start point is not cached in an existing running buffer:

allocating a third running buffer; and

retrieving a portion of the content object not in an existing running buffer as a datastream and inserting the datastream into the third running buffer while delivering the datastream and simultaneously delivering a different part of the content object as a datastream from other existing running buffers.

12. The computer-implemented method of claim 9, further comprising:

determining an advertised length of the content object; and

setting a size of the first buffer as a proportion of the advertised length of the content object.

13. The computer-implemented method of claim 9, further comprising:

analyzing a frequency of requests for the content object; and

modifying a size of the first buffer in response to the analyzing of the frequency of requests for the content object in order to optimize an allocation of a memory.

14. The computer-implemented method of claim 9, further comprising:

prior to allocating a buffer, checking if a sufficient memory for allocating the buffer is available; and

if the sufficient memory is not available, applying a replacement algorithm to reclaim buffers from less frequently requested objects.

IX. Evidence Appendix

No evidence is herein appended.

X. Related Proceedings Appendix

No related proceedings.